SATURATED ARITHMETIC IN A PROCESSING UNIT

BACKGROUND OF THE INVENTION

Technical Field of the Invention

[0001] The present invention relates generally to performing saturated arithmetic in a processing unit.

Background Information

[0002] Saturated arithmetic operations are of increasing importance to numerous types of processing applications such as fixed-point algorithms used for media and signal processing applications. Typically, these applications are computationally intensive, executing millions of saturated operations in a second. Unlike a standard arithmetic operation, saturated arithmetic operations are designed to effectively handle instances in which computations overflow or underflow the data type (e.g., 4-bit twos complement) of the operation. For exemplary purposes, consider the unsigned 4-bit binary addition computation below. The computation on the left is performed with an unsaturated addition operation, and the computation on the right is performed with a saturated addition operation:

Unsaturated Example	**Saturated Example**
1101	1101
+ 1110	+1110
(1) 1011 (Overflow)	1111

[0003] The result of the unsaturated operation produces an overflow (i.e., the correct answer cannot fit into a four bit unsigned data structure). However, the saturated operation produces the highest possible value of a four bit unsigned data structure (i.e., 1111). Instead of producing an overflow, a saturated operation "saturates" the result to a maximum (or minimum) possible value. More specifically, when the computation produces a result larger than the maximum value, the maximum value becomes the result of the saturated operation. Conversely, when the computation produces a result smaller than the minimum value, the minimum value becomes the result of the saturated operation.

[0004] Typically, saturated arithmetic operations are incorporated into the instruction set by adding unique saturated arithmetic instructions. For example, an instruction set may include saturated and unsaturated versions of an integer addition instruction (IADD). Low-level code (e.g., assembly code) may directly use these saturated instructions to perform saturated operations. However, a compiler that compiles high-level code (e.g., Java, C) may be unaware of the saturated instructions in the instruction set. Therefore, a modified compiler may be needed to compile high-level code to object code that utilizes the saturated instructions. The ability to perform saturated arithmetic while avoiding the problems and complications associated with modifying an instruction set is desirable.

BRIEF SUMMARY

[0005] In some embodiments a system comprises an overflow control bit, a programmable saturation control bit, a processing unit, and a saturation unit coupled to the processing unit. A selection unit may select the output of the processing unit or the output of the saturation unit based on the state of the saturation control bit. Further, the saturation control unit may output a saturated or unsaturated value based on the overflow control bit.

NOTATION AND NOMENCLATURE

[0006] Certain terms are used throughout the following description and claims to refer to particular system components. As one skilled in the art will appreciate, various companies may refer to a component by different names. This document does not intend to distinguish between components that differ in name but not function. In the following discussion and in the claims, the terms "including" and "comprising" are used in an open-ended fashion, and thus should be interpreted to mean "including, but not limited to...." Also, the term "couple" or "couples" is intended to mean either an indirect or direct connection. Thus, if a first device couples to a second device, that connection may be through a direct connection, or through an indirect connection via other devices and connections. The term "system" is used to refer to a collection of components. For example, a system may comprise a processor and memory and other components. A system also may comprise a collection of components internal to a single processor and, as such, a processor may be referred to as a system.

BRIEF DESCRIPTION OF THE DRAWINGS

[0007] For a more detailed description of the preferred embodiments of the present invention, reference will now be made to the accompanying drawings, wherein:

[0008] Figure 1 shows a diagram of a processor in accordance with preferred embodiments of the invention; and

[0009] Figure 2 shows a timing diagram of saturated arithmetic operation in accordance with preferred embodiments of the invention.

DETAILED DESCRIPTION OF THE PREFERRED EMBODIMENTS

[0010] The following discussion is directed to various embodiments of the invention. Although one or more of these embodiments may be preferred, the embodiments disclosed should not be

interpreted, or otherwise used, as limiting the scope of the disclosure, including the claims, unless otherwise specified. In addition, one skilled in the art will understand that the following description has broad application, and the discussion of any embodiment is meant only to be exemplary of that embodiment, and not intended to intimate that the scope of the disclosure, including the claims, is limited to that embodiment.

[0011] In accordance with the preferred embodiment, saturated instructions are not added to the instruction set of the processor. Instead, a processor can be selectively configured for execution of saturated or non-saturated operations as discussed below.

[0012] Figure 1 shows an exemplary block diagram of a processor 100. As shown, the processor 100 includes a plurality of registers 102, an arithmetic logic unit ("ALU") 104, a multiplier 106, a saturation unit 108, and a control register 110. Although not explicitly shown, these components preferably reside in the "core" of processor 100. In addition, processor 100 may include other components such as a decode logic, an instruction fetch logic, an address generation unit, and a multiply and accumulate unit. Further, different architectures besides that shown in Figure 1 are possible.

[0013] An operation may begin by retrieving operands from the registers 102 and processing the operands to ALU 104. Alternatively, operands to ALU 104 may be retrieved from external memory and/or a data storage unit (not specifically shown). After processing, the output of ALU 104 is sent to saturation unit 108 and multiplexer 106. The multiplexer 106 preferably selects one of its two inputs to store into registers 102. The first input of the multiplexer 106 is the unsaturated output of the ALU 104 and the second input is the saturated output of the saturation unit 108. The input selected by multiplexer 106 is stored into registers 102 and becomes the result of the operation.

[0014] Control register 110 may be one of the registers 102 and comprises a plurality of fields including at least two fields 112 and 114 that are useful to control the operation of saturation unit 108 and multiplexer 106. An "overflow flag" field 112 (OVFW) indicates whether an overflow (or underflow) occurred in the operation performed by the ALU 104. The overflow flag 112 is controlled by the ALU 104 and is preferably set when the ALU 104 detects an overflow in the operation being performed. A "saturation control flag" field 114 enables and disables the saturation functionality described herein. When enabled, the saturation control flag 114 causes arithmetic instructions to produce saturated results. When disabled, the saturation control flag 114 causes arithmetic instructions to produce unsaturated results. The saturation control flag 114 thus can be used to selectively configure the processor 100 for saturated or unsaturated operands, thereby avoiding the need to have saturated and unsaturated versions of various instructions. An application programming interface (API) may be used to set the state (i.e., enabled or disabled) of the saturation control flag 114. The saturation control flag 114 may also be set by the compiler through a specific pragma, or comment line in the high-level code, or directly set in an assembly language macro that is instantiated in the high-level code.

[0015] Saturation unit 108 preferably receives the output of ALU 104 and performs a saturation process on this output. The saturation process effectively saturates an input to the highest (or lowest) possible value the data type will allow if the overflow flag 112 indicates that the operation performed by ALU 104 produced an overflow (or underflow). If no overflow/underflow conditions exist, the saturation unit 108 outputs the same value that was received as an input to the saturation unit. Thus, if an overflow/underflow does not occur, saturation unit 108 outputs the value it received from ALU 104.

[0016] Since the instruction set of processor 100 may not contain saturated arithmetic instructions, a program developer may call a function within the API to perform saturated operations if desired. Alternatively, saturated operations may be performed after the compiler sets the saturation control flag 114 through a specific pragma, or comment line in the high-level code, or the saturation control flag 114 is directly set in an assembly language macro that is instantiated in the high-level code.

[0017] The function that allows saturated arithmetic to be performed may be referred to as a "saturation control function". Based upon the saturation control flag 114, multiplexer 106 may select either the output of saturation unit 108 or the output of ALU 104 to store in registers 102. When a program developer desires to use a saturated arithmetic operation, the saturation control function preferably is executed to enable the saturation control flag 114. After the saturation control flag 114 has been enabled, multiplexer 106 will select the output from saturation unit 108 as the result of the operation to store into registers 102. Similarly, when a developer desires to use an unsaturated arithmetic operation, the saturation control function preferably is executed to disable the saturation control flag 114. Disabling the saturation control flag 114 causes the multiplexer 106 to select the output of ALU 104 to be stored in registers 102. The saturation control flag 114 preferably remains in its programmed state until reprogrammed through the saturation control function, thereby permitting extended sequences of instructions to be executed with or without saturated arithmetic.

[0018] Figure 2 shows an exemplary timing diagram of a saturated addition operation. Four cycles of a CPU clock 200 associated with processor 100 are shown to aid in the discussion. Typically, all unsaturated arithmetic instructions (e.g., addition, subtraction) require one execution cycle. However, a multiply and accumulate (MAC) unit may require two execution cycles to execute a

multiply instruction. As shown in Figure 2, two multiply instructions 202 and 204 may be executed consecutively in a MAC unit. In parallel with the second multiply instruction 204, a saturated addition may be performed. As explained below, figure 2 illustrates that, in some situations, saturated operations may not introduce extra latency.

[0019] In accordance with the preferred embodiment, while saturation control bit 114 is enabled, all arithmetic instructions require one additional execution cycle. During the first cycle 210, ALU 104 preferably performs the unsaturated ADD instruction 206 contained within the instruction set of processor 100. If an overflow/underflow occurs while performing this instruction, the overflow flag 112 accordingly may be updated by the ALU 104 during this first cycle 210. During the second cycle 212, the saturation process 208 (SAT) preferably occurs via the operation of the saturation unit 18, which accepts an unsaturated input from ALU 104 and produces a saturated output. If the overflow flag 112 indicates that an overflow/underflow occurred in ALU 104 during cycle 210, the saturation process saturates the input. If the overflow flag 112 indicates that an overflow/underflow did not occur in ALU 104, the output from the saturation unit 108 is made identical to the input (not saturated).

[0020] To perform unsaturated arithmetic operations, saturation control bit 114 is disabled by action of the saturation control function. While the processor 100 is in unsaturated mode, the multiplexer 106 selects the unsaturated output of ALU 104 as the result of operations and ignores the output of the saturation unit 108.

[0021] Although ALU 104 is used in the preferred embodiment, any suitable type of processing unit (e.g., a multiplier and accumulate unit) similarly may be used as input to the saturation unit 108, thereby allowing other saturated operations to occur, such as left shifts.

[0022] While the preferred embodiments of the present invention have been shown and described, modifications thereof can be made by one skilled in the art without departing from the spirit and teachings of the invention. The embodiments described herein are exemplary only, and are not intended to be limiting. Many variations and modifications of the invention disclosed herein are possible and are within the scope of the invention. Accordingly, the scope of protection is not limited by the description set out above. Each and every claim is incorporated into the specification as an embodiment of the present invention.